Bit-Precise Neural Network Verification

Edoardo Manino

The University of Manchester

15 January 2025

EnnCore

The University of Manchester

Today

Why this presentation?

- A microcosm of spec debates
- What SONNX can be used for

Agenda

- Brief introduction
- NN verification
- C language for float NN spec
- Performance of software verifiers
- Issues and future work

Feel free to ask questions!

Systems and Software Security (S3) Group

L. Cordeiro
(Software Security)

R. Banach
(Formal Methods)

M. Mustafa
(Applied Cryptography)

N. Zhang
(IoT Security)

E. Manino
(AI Security)

B. Magri
(Cryptography)

Y. Sun
(Trustworthy AI)

D. Dresner
(System Governance)

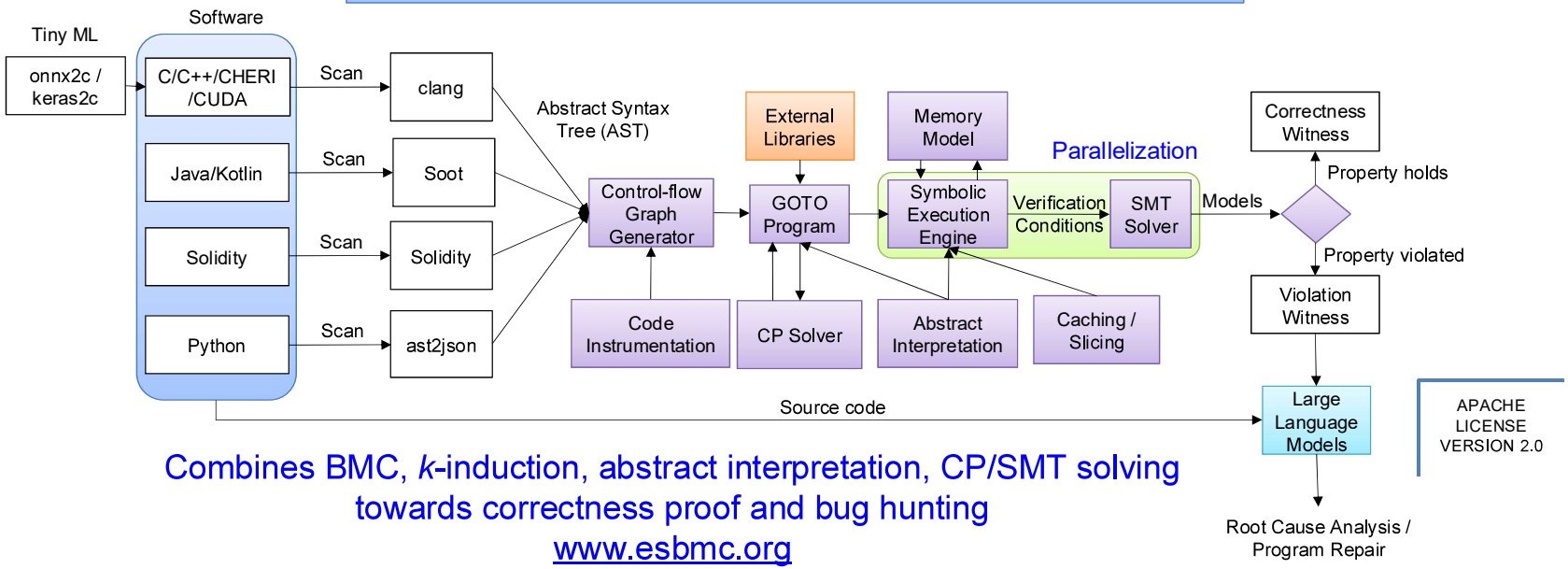
A. Creswell
(cyber and AI)

G. Smith
(Former director of GCHQ)

- Part of the Computer Science department at UoM
- Academic Centre of Excellence in Cyber Security Research

ESBMC: A Logic-based Verification Platform

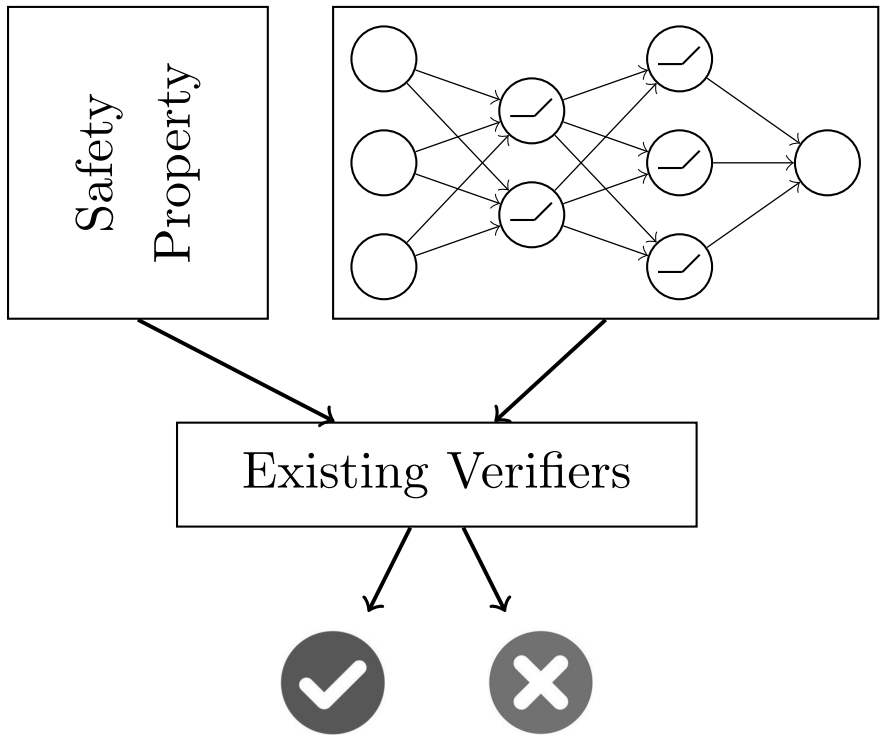
Logic-based automated reasoning for verifying the **safety** and **security** of **AI and software systems**



Combines BMC, *k*-induction, abstract interpretation, CP/SMT solving towards correctness proof and bug hunting

www.esbmc.org

Background: neural network verification



Mainstream approach (à la VNN-COMP)

- Neural network in high-level format (ONNX, PyTorch. . .)
- "Simple" property in FOL fragment (pre- and post-conditions)

Background: safety property example

Robustness property

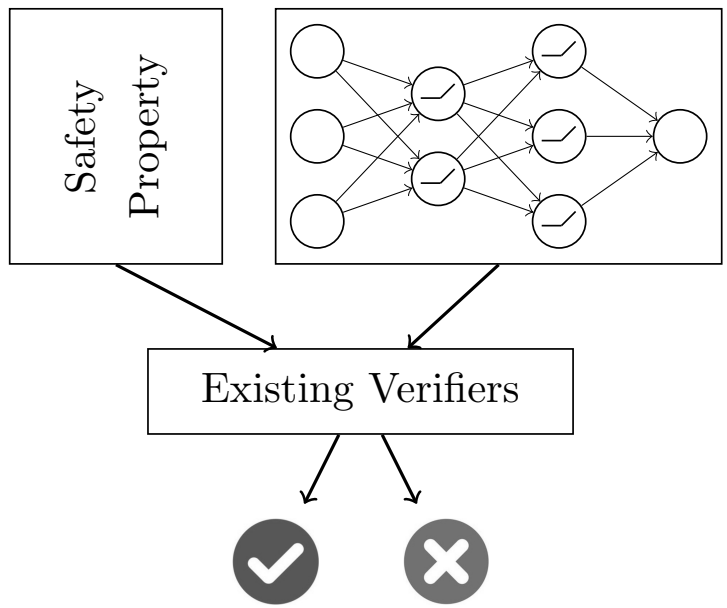
- Pick an input x and a classifier f
- $\forall x', ||x - x'|| < \epsilon \implies f(x) = f(x')$

"Classic ML" mindset

- $f : \mathbb{R}^n \to \mathbb{R}^m$
- VNN-COMP assumption

"Real-world" networks

- $f : \mathbb{F}^n \to \mathbb{F}^m$
- Or even $f : \mathbb{Z}^n \to \mathbb{Z}^m$



Implementation effects (1)

Can we expect consistent behaviour across devices?

- Cidon et al., *Characterizing and taming model instability across edge devices*, 2021
- Wang et al, *SysNoise: exploring and benchmarking trainin-deployment system inconsistency*, 2023
- Schlögl et al., *Causes and Effects of Unanticipated Numerical Deviations in Neural Network Inference Frameworks*, 2023

Many low-level sources of noise!

- Pre-processing: .jpg→tensor (iDCT, interpolation, colour)
- Model inference: convolutions, upsampling, floats, quantize
- Post-processing: tensor→bounding box (rounding coordinates)

Up to 6% accuracy fluctuation[1]

[1] Cidon [2021] runs MobileNetV2 on photos taken from five different phones.

Implementation effects (2)

Can we trust NN verifiers?

- VNN-COMP compares the best neural network verifiers
- Let's reproduce one of their results!

Benchmark: reach_prob_density/robot_11

- A ReLU network with architecture $5 \times 64 \times 64 \times 64 \times 5$
- Input assumption: $x_0 \in [-1.8, -1.2] \wedge x_1 \in [-1.8, -1.2] \ldots$
- Output assertion: $y_0 \geq 0.27 \wedge y_1 \in [-0.17, 0.17] \ldots$

Five tools return a counterexample!

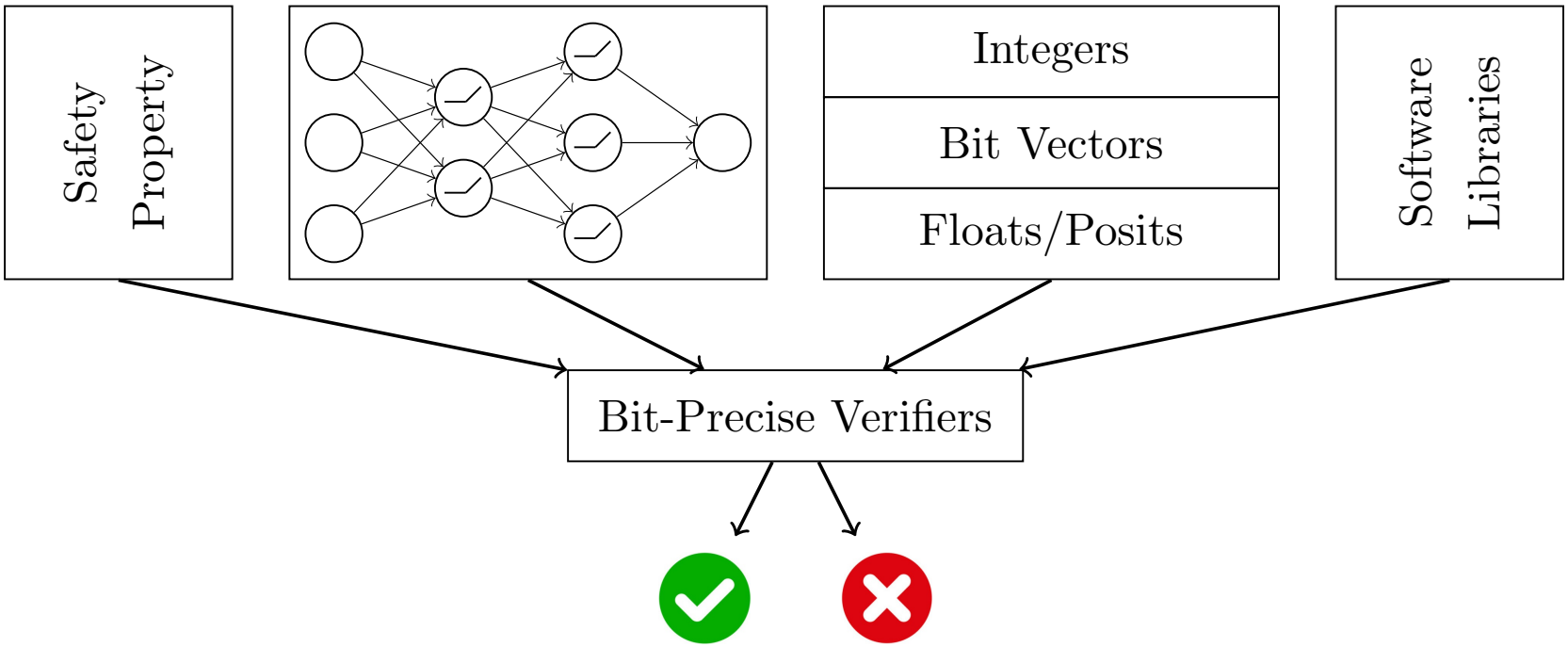
- $\alpha\beta$-CROWN, Marabou, `nnenum`, VeriNet, Peregrinn

But none of them violates the output assertion[2]

[2] With the plain C code from `onnx2c` and the MinGW-w64 compiler.


YOU PROVED THAT A ML MODEL IS SAFE
YOUR GUARANTEE IS IMPLEMENTATION DEPENDENT
imgflip.com

Bit-precise neural network verification



We need more precision!

- Bit-precise machine arithmetic (float or integer)
- Exact order of operations (requires knowledge of software)
- Guarantees sound proofs (unless the hardware is misbehaving)


Develop
yet
another software
verifier
Check
the performance
of
existing ones
imgflip.com

NeuroCodeBench (1)

Neural network specification

- Produce microcontroller-style C code
- Use `keras2c`, `onnx2c` for translation

Safety property specification

- Add explicit pre- and post-conditions in the code
- E.g. `assert(x >= 0.4287175f);`

Benchmarking goals

- Compatibility: follow SV-COMP conventions
- Scalability: from small to "large" instances
- Correctness: verdicts must be known (do not rely on SV)

NeuroCodeBench (2)

Benchmark Category	Safe	Unsafe	Ground Truth
`math_functions`	33	11	A Priori
`activation_functions`	40	16	A Priori
`hopfield_nets`	47	33	A Priori
`poly_approx`	48	48	Brute Force
`reach_prob_density`	22	13	VNN-COMP'22
`reinforcement_learning`	103	193	VNN-COMP'22
Total	293	314	

Table: Overview of *NeuroCodeBench*. The "Unsafe" column comprises all properties for which a counterexample exists. The "Ground Truth" column reports the source of our verdicts.

NeuroCodeBench (3)

Functions from `math.h`

- expf, expm1f, logf, log1pf
- acosf, asinf, atanhf, cosf, sinf, tanhf
- erff, fabsf, fmaxf, sqrtf

Activation functions

- Elu, Gelu, Logistic, ReLU, Softmax, Softplus, Softsign, TanH

Safety properties (Examples)

- Output bounds: $expf(x) \geq 1 + x$
- Periodicity: $sinf(x) = sinf(x + 2\pi)$
- Symmetry: $tanhf(x) = -tanhf(-x)$
- Inversion: $expf(logpf(x)) = x$

Benchmark Category
math_functions
activation_functions
hopfield_nets
poly_approx
reach_prob_density
reinforcement_learning

```
1   #include <verifier_functions.h>
2
3   #include <math.h>
4
5   float elu(float x)
6   {
7           if(x >= 0.0f)
8                   return x;
9           else
10                  return expm1f(x);
11  }
```

NeuroCodeBench (4)

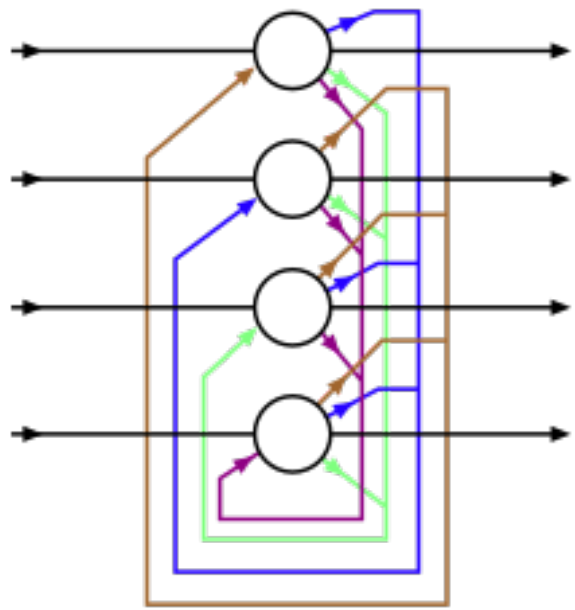
Classic Hopfield Networks

- Recurrent architecture
- Hard-coded Hebbian weights
- Error-correcting behaviour

Benchmark Category
`math_functions`
`activation_functions`
`hopfield_nets`
`poly_approx`
`reach_prob_density`
`reinforcement_learning`

Our idea

- Reconstruct a single $x = (1, 1, \ldots, 1)$
- Use either softsign or tanh activations
- Vary code width and num of iterations



Safety properties

- Make $x_i \in [-1, +1]$ for $i <$ half width
- Can the network reach $x = (1, \ldots, 1)$?

NeuroCodeBench (5)

Transfer function approximation

- Very common in engineering
- Approximate electrical equivalent

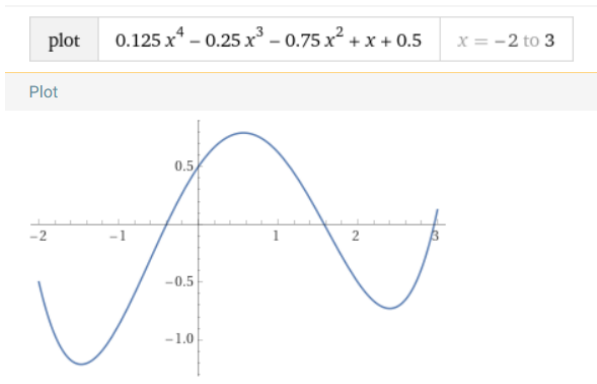
Our idea

- Fourth-order oscillating polynomial
- ReLUs, 1–4 layers, 16–1024 width

Safety properties

- Robustness of approximation
- $|\text{network}(x) - \text{poly}(x)| \leq \epsilon$
- for x around the worst-case

Benchmark Category
`math_functions`
`activation_functions`
`hopfield_nets`
`poly_approx`
`reach_prob_density`
`reinforcement_learning`



NeuroCodeBench (6)

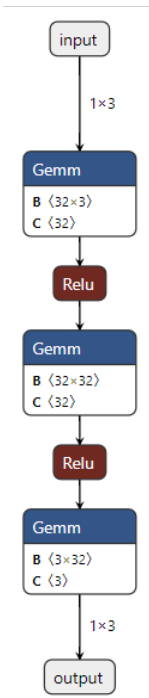
Importing VNN-COMP benchmarks

- Choose categories from 2022 edition
- with very small neural networks

Benchmark Category
`math_functions`
`activation_functions`
`hopfield_nets`
`poly_approx`
`reach_prob_density`
`reinforcement_learning`

Converting to plain C code

- Turn ONNX into plain C with `onnx2c`
- Microcontroller-style minimalism



Safety properties

- Keep the original VNN-LIB properties
- Encode them as `assert`/`assume`
- Check validity of counterexamples

NeuroCodeBench (7)

Benchmark Category	Safe	Unsafe	Ground Truth
`math_functions`	33	11	A Priori
`activation_functions`	40	16	A Priori
`hopfield_nets`	47	33	A Priori
`poly_approx`	48	48	Brute Force
`reach_prob_density`	22	13	VNN-COMP'22
`reinforcement_learning`	103	193	VNN-COMP'22
Total	293	314	

Table: Overview of *NeuroCodeBench*. The "Unsafe" column comprises all properties for which a counterexample exists. The "Ground Truth" column reports the source of our verdicts.


COMPETITION
TIME!

A short story (1)

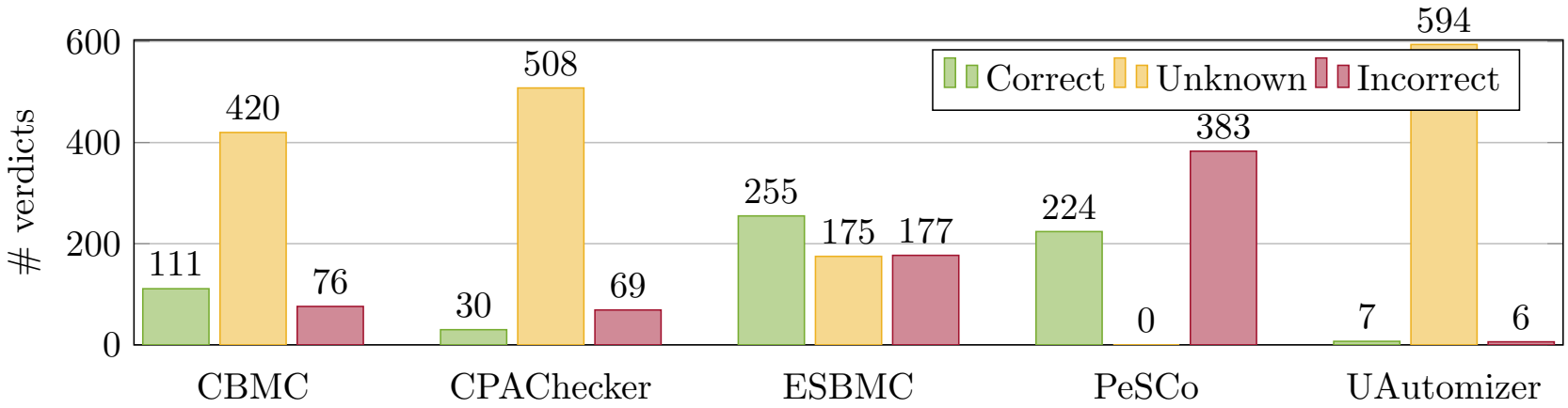


Figure: top software verifiers after 900 seconds **(August 2023)**.

Experiments with off-the-shelf verifiers

- We pick the top scoring tools from SV-COMP 2022
- We keep the same settings of the reachability category
- Some of these tools have competed for **decades**
- Variety of techniques: BMC, automata, portfolios

A short story (2)

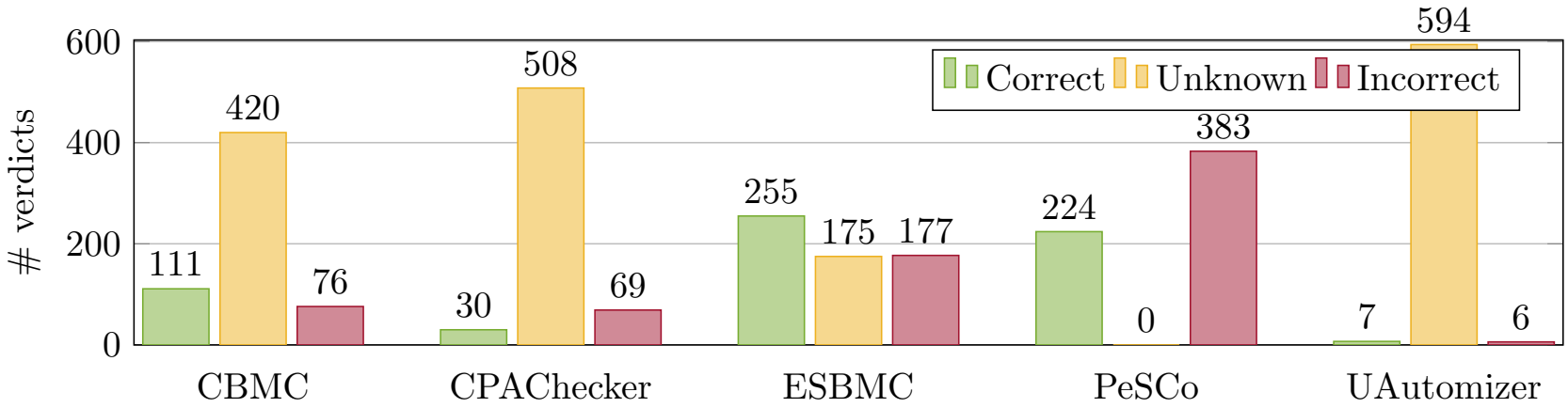


Figure: top software verifiers after 900 seconds **(August 2023)**.

Our opinion

- No support of mathematical libraries → incorrect results
- Cannot scale to large programs → unknown result (timeout)

Is there anything else at play here?


You discover that verifiers are broken
THIS IS FINE.
But that means more future work!
imgflip.com

A short story (3)

Reproducibility goal

- Submit *NeuroCodeBench* to SV-COMP 2023
- Experiments run by independent team
- Tool authors have a chance to fix bugs

Community engagement (October 2023)

- After some discussion[3] *NeuroCodeBench* is approved
- All future editions of SV-COMP will use it

Improve ESBMC (November 2023)

- At Manchester, we develop one of the top software verifiers
- *NeuroCodeBench* is breaking our own tool too!

[3] https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/issues/1396

A short story (4)

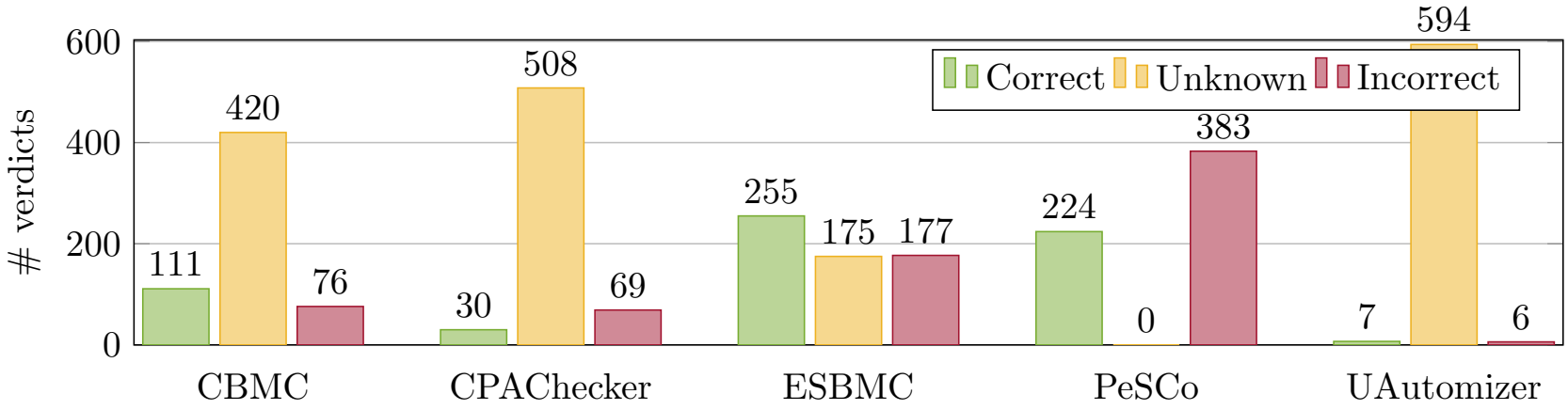


Figure: top software verifiers before SV-COMP **(August 2023)**.

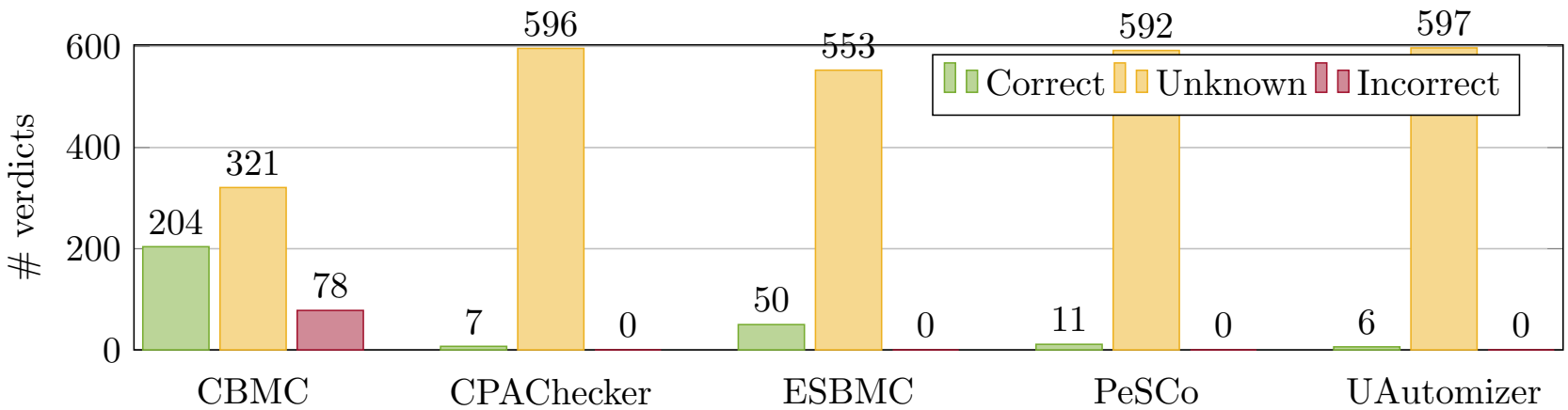


Figure: top software verifiers after SV-COMP **(December 2023)**.

A short story (5)

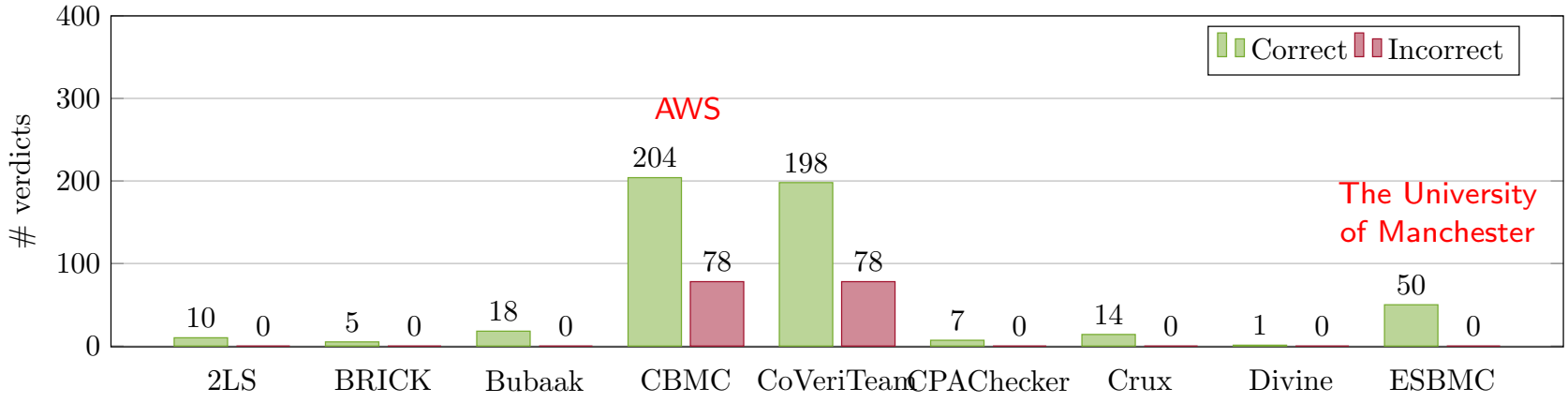


Figure: all software verifiers on *NeuroCodeBench* **(December 2023)**.

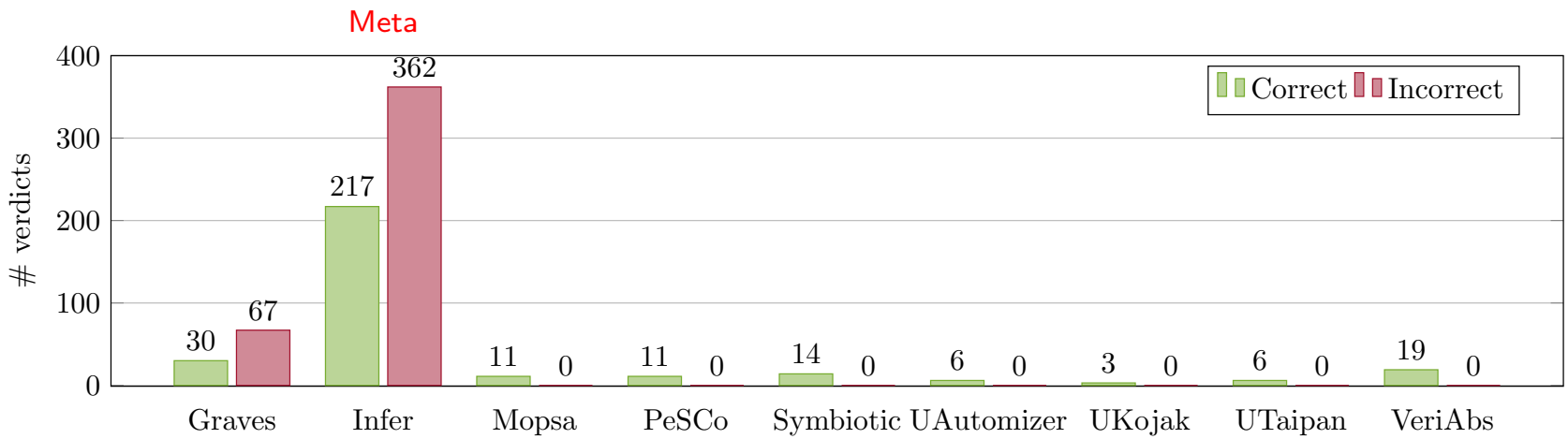


Figure: all software verifiers on *NeuroCodeBench* **(December 2023)**.

or so they thought. . .

Summary

Short benchmark paper

- *NeuroCodeBench: a plain C neural network benchmark for software verification*. AFRiTS workshop @ SBMF 2023.
- `https://arxiv.org/abs/2309.03617`
- `https://github.com/emanino/plain_c_nn_benchmark`
- Extended (journal) version in a few months

Position paper on PL, NN & verification

- *Neural Network Verification is a Programming Language Challenge*. Accepted @ ESOP 2025.
- `https://arxiv.org/abs/2501.05867`

Any questions?